

Finding minimum Tucker submatrices

Ján Maňuch^{1,2} and Arash Rafiey²

¹ Department of Computer Science, UBC, Vancouver, BC, Canada

² Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada
jmanuch@cs.ubc.ca, arashr@sfu.ca

Abstract. A binary matrix has the Consecutive Ones Property (C1P) if its columns can be ordered in such a way that all 1s on each row are consecutive. These matrices are used for DNA physical mapping and ancestral genome reconstruction in computational biology on the other hand they represent a class of convex bipartite graphs and are of interest of algorithm graph theory researchers. Tucker gave a forbidden submatrices characterization of matrices that have C1P property in 1972. Booth and Lucker (1976) gave a first linear time recognition algorithm for matrices with C1P property and then in 2002, Habib, et al. gave a simpler linear time recognition algorithm. There has been substantial amount of works on efficiently finding minimum size forbidden submatrix. Our algorithm is at least n times faster than the existing algorithm where n is the number of columns of the input matrix.

1 Introduction and Preliminaries

A binary matrix has the Consecutive Ones Property (C1P) if its columns can be ordered in such a way that all ones in each row are consecutive. Deciding if a matrix has the C1P can be done in linear-time and space [4,7,8,12,13]. The problem of deciding if a matrix has the C1P has been considered in genomic, for problems such as physical mapping [2,9] or ancestral genome reconstruction [1,5,11].

Let M be a $m \times n$ binary matrix. Let $\mathbf{R} = \{r_i : i = 1, \dots, m\}$ be the set of its rows and $\mathbf{C} = \{c_j : j = 1, \dots, n\}$ the set of its columns. Its *corresponding bipartite graph* $G(M) = (V_M, E_M)$ is defined as follows: $V_M = \mathbf{R} \cup \mathbf{C}$, and two vertices $r_i \in \mathbf{R}$ and $c_j \in \mathbf{C}$ are connected by an edge if and only if $M[i, j] = 1$. We will refer to the partition \mathbf{R} and \mathbf{C} of $G(M)$ as black and white vertices, respectively. The set of neighbors of a vertex x will be denoted by $N(x)$. The i -neighborhood of x , denoted by $N_i(x)$, is the set of vertices distance i from x . All these sets, for a fixed x , can be computed in time $O(e)$ using the bread-first search algorithm. A subgraph of $G(M)$ induced by vertices x_1, \dots, x_k will be denoted by $G(M)[x_1, \dots, x_k]$. A set of edges of bipartite graph is called *induced matching* if the set of endpoints of these edges induces this matching in the graph. For example, two edges $\{u, v\}$ and $\{u', v'\}$, where u, u' are in the same partition form an induced matching if $\{u, v'\}$ and $\{u', v\}$ are not edges of the graph.

An *asteroidal triple* is an independent set of three vertices such that each pair is connected by a path that avoids the neighborhood of the third vertex. A *white asteroidal triple* is an asteroidal triple on white (column) vertices.

The following result of Tucker links the C1P of matrices to asteroidal triples of their bipartite graphs.

Theorem 1 ([14]). *A binary matrix has the C1P if and only if its corresponding bipartite graph does not contain any white asteroidal triples.*

Theorem 2 ([14]). *A binary matrix has the C1P if and only if its corresponding bipartite graph does not contain any of the forbidden subgraphs in $T = \{G_{I_k}, G_{II_k}, G_{III_k} : k \geq 1\} \cup \{G_{IV}, G_V\}$, depicted in Figure 1. We will refer to these subgraphs as the type I, II, III, IV and V, respectively.*

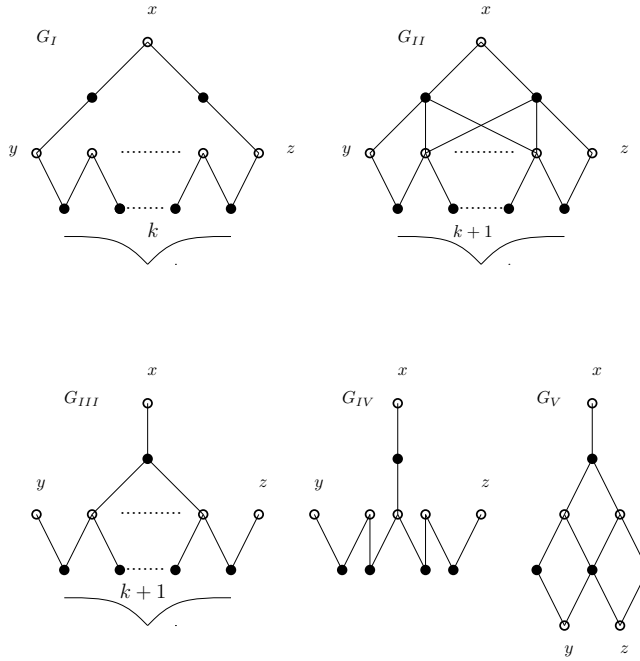


Fig. 1. The set of Tucker's forbidden subgraphs.

The author in [10] developed an algorithm for finding one of the obstructions in linear time. However, their algorithm does not guarantee the minimum size obstruction. The characterization can be used to determine whether a given binary matrix has the C1P in time $O(\Delta mn^2 + n^3)$, where Δ is the maximum number of ones per row, i.e., the maximum degree of black vertices in $G(M)$, as explained by the following result in [6].

Lemma 1 ([6]). *A white asteroidal triple u, v, w with the smallest sum of the three paths (avoiding the third neighborhood) can be computed in time $O(\Delta mn^2 + n^3)$.*

For practical purposes, there is a much faster algorithm that uses PQ-trees for determining whether a binary matrix has the C1P, cf. [4]. Tucker's interest was in finding the smallest submatrix of a non-C1P binary matrix which makes this matrix non-C1P. He further refined his asteroidal triple characterization using a set of *forbidden submatrices*. We will state this results in terms of *forbidden subgraphs*.

We will consider two problems: (1) detecting a smallest forbidden subgraph of each type (Section 2), and (2) detecting a smallest forbidden subgraph of any type (Section 3).

We use the followings to improve the complexity :

- In our computation we use degree of each vertex instead the maximum degree Δ .
- We compute some of the necessary sets in advance.
- In our analysis we use the minimum obstruction assumption and explore the connection of vertices around a minimum obstruction with it.

Subgraph type	Time complexity		
	Previous result	Our result (Exact)	Our result
I	$O(\Delta^4 m^3)$ [3]	$O(\Delta e^2) = O(\Delta^3 m^2)$	$O(n^2 e)$ [6]
II	$O(\Delta^4 m^3) = O(ne^3)$ [3]	—	$O(n^2 e)$ [6]
III	$O(\Delta^2 m^2 n^2)$ [3]	$O(e^3) = O(\Delta^3 m^3)$	$O(ne^2)$
IV	$O(\Delta^3 m^2 n^3)$ [6]	$O(m^3 e) = O(\Delta m^4)$	$O(n^3 e)$
V	$O(\Delta^4 m^2 n)$ [6]	$O(m^3 e) = O(\Delta m^4)$	$O(n^3 e)$
Any	$O(\Delta^3 m^2 (\Delta m + n^3))$	$O(ne(n^2 + e)) = O(\Delta mn(\Delta m + n^2))$	

Table 1. Comparison of our results with the previous results.

Note that without loss of generality we can assume that M does not contain any all-zero columns or rows, as such columns does not affect whether the matrix has the C1P or the forbidden submatrices of M . It follows that $\Delta m \geq n$. We will use this assumption throughout this paper. Also note that the number of edges in $G(M)$ is the same as the number of ones in M , which we denote as e . Note that $e = O(\Delta m)$ and that $e \geq m, n$ (since we assume that there are no all-zero columns or rows in M).

We will use the following auxiliary lemma.

Lemma 2. *Given a bipartite graph G with e edges and partitions of size m and n , picking an induced matching of size two of G or determining that no such induced matching exists can be done in time $O(e + m + n)$.*

Proof. Let U be the partition of size n . Order vertices of U by their degrees: $\deg(u_1) \leq \deg(u_2) \leq \dots \leq \deg(u_n)$. For every $i = 1, \dots, n-1$, check if $N(u_i) \setminus N(u_{i+1})$ is non-empty. If for some i , $N(u_i) \setminus N(u_{i+1}) \neq \emptyset$, then also $N(u_{i+1}) \setminus N(u_i) \neq \emptyset$. In this case, we can pick any $a \in N(u_i) \setminus N(u_{i+1})$ and any $b \in N(u_{i+1}) \setminus N(u_i)$, and return $\{u_i, a\}$ and $\{u_{i+1}, b\}$, as it forms an induced matching of G .

Now, assume that for every i , $N(u_i) \setminus N(u_{i+1}) = \emptyset$, i.e., $N(u_i) \subseteq N(u_{i+1})$. We will show that there is no induced matching of G of size two. Assume for contradiction that $\{u_i, a\}$ and $\{u_j, b\}$, where $i < j$, is such an induced matching. We have $N(u_i) \subseteq N(u_{i+1}) \subseteq \dots \subseteq N(u_j)$, i.e., $a \in N(u_j)$, a contradiction. Hence, in this case we can report that there is no such matching.

Vertices of U can be sorted by their degrees in time $O(n + m)$ using a count sort. For each i , checking if $N(u_i) \setminus N(u_{i+1})$ is non-empty can be done in time $O(\deg(u_i))$, hence, the total time spent on checking is $O(\sum_{i=1}^{n-1} \deg(u_i)) = O(e)$.

2 Detection of smallest forbidden subgraphs for each type

We will present four algorithms which find a smallest subgraph of type I, III, IV and V, respectively, each improving the complexity of the best known such algorithm, cf. [3]. For type II, we refer reader to the $O(ne^3)$ algorithm³ in [3].

2.1 Type I

Algorithm 1 finds a smallest forbidden subgraph of type I in time $O(\Delta e^2)$.

Algorithm 1: Find a smallest G_{I_k} subgraph.

Input : $G(M)$
Output: A smallest subgraph G_{I_k} of $G(M)$

```

1 for  $w \in \mathbf{R}$  do
2   for  $x, y \in N(w)$  do
3     construct the subgraph  $G_{w,x,y}$  of  $G(M)$  induced by vertices
4        $(\mathbf{R} \setminus (N(x) \cap N(y))) \cup (\mathbf{C} \setminus N(w)) \cup \{x, y\}$ ;
5     find a shortest path between  $x$  and  $y$  in  $G_{w,x,y}$ ;
6     if the length of the path is smaller than any observed so far then
7       remember  $w$  and the vertices of the path;
8     end
9   end
10 return subgraph of  $G(M)$  induced by the remembered set of vertices (if any)
```

Correctness of Algorithm 1. We are looking for induced cycles of length 6 or more. For each black vertex w and its two neighbors x, y , we find a shortest

³ The authors of [3] showed that the complexity of their algorithm is $O(\Delta^4 m^3)$, however, it is easy to check that their algorithm works in time $O(ne^3)$.

induced cycle of length at least 6. Such cycle cannot contain any vertex incident with w other than x and y , and any vertex incident with both x and y other than w . Hence, a shortest such cycle c can be obtained from the a shortest $x - y$ path p in $G_{w,x,y}$ by adding two edges $\{x, w\}$ and $\{y, w\}$. This cycle cannot be of length 4, otherwise p would contain a vertex in $N(x) \cap N(y)$. It remains to show that c is induced. Assume that there is a chord $\{u, v\}$ in c . Since p does not contain $N(w) \setminus \{x, y\}$, $u, v \neq w$. Hence, we could use the chord as a shortcut to find a shorter cycle containing edges $\{x, w\}$ and $\{y, w\}$, and hence, a shorter path between x and y in $G_{w,x,y}$, a contradiction.

Complexity of Algorithm 1. We will show that the complexity of Algorithm 1 is $O(\Delta e^2) = O(\Delta^3 m^2)$. The first loop executes m times and the second $\deg(w)^2$ times. Hence, the body of the second loop executes $\sum_{w \in \mathbf{R}} \deg(w)^2 = O(\Delta e)$ times. Constructing graph $G_{w,x,y}$ takes time $O(e)$ and finding a shortest path in $G_{w,x,y}$ can be done in time $O(e)$ using the Breadth-first search algorithm.

2.2 Type III

Algorithm 2 finds a smallest forbidden subgraph of type II in time $O(e^3)$.

Algorithm 2: Find a smallest G_{III_k} subgraph.

```

Input   :  $G(M)$ 
Output : A smallest subgraph  $G_{\text{III}_k}$  of  $G(M)$ 

1 for  $\{x, w\} \in E_M$ , where  $x \in \mathbf{C}$  and  $w \in \mathbf{R}$  do
2   for  $\{y, a\}$ , where  $y \in \mathbf{C} \setminus N(w)$  and  $a \in \mathbf{R} \setminus N(x)$  do
3     construct the subgraph  $G_{x,w,y,a}$  of  $G(M)$  induced by vertices
        $(\mathbf{R} \setminus N(x) \setminus N(y)) \cup \{a\} \cup (N(w) \setminus \{x\}) \cup (\mathbf{C} \setminus N(a))$ ;
4     find a shortest path between  $a$  and set  $\mathbf{C} \setminus N(w) \setminus N(a)$  in  $G_{x,w,y,a}$ ;
5     if if the path exists and is shorter than any observed so far then
6       | remember  $w, x, y$  and the path;
7     end
8   end
9 end
10 return subgraph of  $G(M)$  induced by the remembered set of vertices (if any)

```

Correctness of Algorithm 2. Let us first verify that the vertices of a shortest path found in line 4 and w, x, y induce a subgraph of type III. Obviously, x is connected only to w , w is not connected to y and the last vertex z of the path. On the other hand, w must be connected to all other white vertices on the path, since any such white vertex that is not in $N(w)$ is in $\mathbf{C} \setminus N(a)$ and hence, also $\mathbf{C} \setminus N(w) \setminus N(a)$, i.e., we would have a shorter path ending at this vertex. Since the path is a shortest path, all black vertices on the path are connected only to its predecessor and successor on the path. In addition a is connected to y and no other black vertex on the path is connected to y since $G_{x,w,y,a}$ does not contain any other neighbors of y . It follows that the vertices w, x, y and the vertices of a shortest path induce a subgraph of type III.

Second, consider a smallest subgraph of type III in $G(M)$. We will show it is considered by the algorithm. Assume the algorithm is in the cycle, where it

picked edges $\{x, w\}$ and $\{y, a\}$ of this subgraph. Then the rest of the vertices must lie in $G_{x,w,y,a}$: the remaining black vertices are not connected to x and y and the remaining white vertices are either in $N(w) \setminus \{x\}$ and z is $\mathbf{C} \setminus N(a)$. These vertices together with a must form a shortest path from a to $\mathbf{C} \setminus N(w) \setminus N(a)$ in $G_{x,w,y,a}$, hence, Algorithm 2 finds this subgraph or a subgraph with the same number of vertices.

Complexity of Algorithm 2. We will show that the complexity of Algorithm 2 is $O(e^3) = O(\Delta^3 m^3)$. The first loop executes e times. The second loop executes $O(e)$ times. Constructing graph $G_{x,w,y,a}$ takes time $O(e)$. Finding a shortest path in G_x can be done in time $O(e)$ using a breadth-first search algorithm.

2.3 Type IV

Algorithm 3 determines if $G(M)$ contains a forbidden subgraph of type IV in time $O(m^3 e)$.

Algorithm 3: Find a G_{IV} subgraph.

Input : $G(M)$
Output: A subgraph G_{IV} of $G(M)$

```

1 for distinct  $a, b, c, d \in R$  do
2   find  $UX = N(a) \setminus (N(b) \cup N(c))$ ;
3   find  $VY = N(b) \setminus (N(a) \cup N(c))$ ;
4   find  $WZ = N(c) \setminus (N(a) \cup N(b))$ ;
5   find  $U = UX \cap N(d)$  and  $X = UX \setminus N(d)$ ;
6   find  $V = VY \cap N(d)$  and  $Y = VY \setminus N(d)$ ;
7   find  $W = WZ \cap N(d)$  and  $Z = WZ \setminus N(d)$ ;
8   if each of the sets  $X, Y, Z, U, V, W$  is non-empty then
9     pick any  $x \in X, y \in Y, z \in Z, u \in U, v \in V, w \in W$ ;
10    return  $G(M)[a, b, c, d, x, y, z, u, v, w]$ 
11  end
12 end
13 return not found

```

Correctness of Algorithm 3. It is easy to see that once a, b, c, d are picked, each of x, y, z, u, v, w has to belong to computed set X, Y, Z, U, V, W , respectively, and that once they are picked from those sets, the returned vertices induce G_{IV} .

Complexity of Algorithm 3. We will show that the complexity of Algorithm 3 is $O(m^3 e) = O(\Delta m^4)$. The time complexity of the steps inside the loop depends on degrees of nodes a, b, c, d , i.e., it is $O(\deg(a) + \deg(b) + \deg(c) + \deg(d))$. Hence, the overall complexity is $\sum_{a,b,c,d \in R} O(\deg(a) + \deg(b) + \deg(c) + \deg(d)) = 4 \sum_{a,b,c,d \in R} O(\deg(d)) = 4 \sum_{a,b,c \in R} O(e) = m^3 e$.

2.4 Type V

Algorithm 4 determines if $G(M)$ contains a forbidden subgraph of type V in time $O(m^3 e)$.

Algorithm 4: Find a G_V subgraph.

Input : $G(M)$
Output: A subgraph G_V of $G(M)$

```
1 for distinct  $a, b, c, d \in \mathbf{R}$  do
2   find  $UY = N(b) \cap N(d) \setminus N(c)$ ;
3   find  $VZ = N(b) \cap N(c) \setminus N(d)$ ;
4   find  $U = UY \cap N(a)$  and  $Y = UY \setminus N(a)$ ;
5   find  $V = VZ \cap N(a)$  and  $Z = VZ \setminus N(a)$ ;
6   find  $X = N(a) \setminus (N(b) \cup N(c) \cup N(d))$ ;
7   if each of the sets  $X, Y, Z, U, V$  is non-empty then
8     pick any  $x \in X, y \in Y, z \in Z, u \in U, v \in V$ ;
9     return  $G(M)[a, b, c, d, x, y, z, u, v]$ 
10  end
11 end
12 return not found
```

Correctness of Algorithm 4. It is easy to see that once a, b, c, d are picked, each of x, y, z, u, v has to belong to computed set X, Y, Z, U, V , respectively, and that once they are picked from those sets, the returned vertices induce G_V .

Complexity of Algorithm 4. The complexity of Algorithm 4 is $O(m^3e) = O(\Delta m^4)$. This follows by the same argument as for Algorithm 4.

3 Detection of a smallest forbidden subgraph

Overall, we will use Dom et al. ([6]) approach to find the smallest forbidden subgraph in $G(M)$. We will first find a shortest-paths (the sum of the lengths of the three paths) white asteroidal triple A in time $O(n^2e) = O(\Delta mn^2)$ using the algorithm in [6].

A shortest-paths white asteroidal triple A must be in T , but does not need to be a smallest forbidden subgraph. Let ℓ be the sum of the lengths of the three paths of A . If A is of

- type I or II, then it contains ℓ vertices;
- type III, it contains $\ell - 5$ vertices;
- type IV, it contains $10 = \ell - 8$ vertices;
- type V, it contains $9 = \ell - 1$ vertices.

It follows that if one of the smallest forbidden subgraphs is of type I or II, then each shortest-paths asteroidal triple is of type I or II and is a smallest forbidden subgraph. For the remaining cases, we need to determine the smallest forbidden subgraphs of type III, IV and V. However, we only need to find a smallest subgraph of type X if it is a smallest forbidden subgraph. Hence, for types IV and V, if we find during the search that there is a smaller forbidden subgraph of some other type, we can stop searching for this type. For type III, since it has a variable size, we cannot stop searching, however, we can abandon the branch which would yield a larger or even the same size subgraph of type III than we have observed. We will use this in what follows to obtain faster algorithms for types III, IV and V than the ones presented in the previous section.

3.1 Type III

Algorithm 5 guarantees to find a smallest subgraph of type III if it is smaller than other types of forbidden subgraphs in time $O(ne^2)$. If there is a smaller subgraph of type I or there is a smaller of same size subgraph of type V in $G(M)$, it either reports that or it could report a subgraph of type III which is not the smallest. It will first determine whether G_{III_1} is a subgraph of $G(M)$. If not it continues to the second phase, where it assumes that the smallest subgraph of type III (if it exists) has at least 9 vertices.

Algorithm 5: Find a smallest G_{III_k} subgraph if it is smaller than other types of subgraphs.

```

Input :  $G(M)$ 
Output: A smallest subgraph  $G_{III_k}$  of  $G(M)$  or report there is a subgraph of other type (I or V) of equal or smaller size

1 for  $w \in \mathbf{R}$  do
2   for  $x, u \in N(w)$  do
3     construct the subgraph  $G_{x,w,u}$  of  $G(M)$  induced by vertices
4        $N(u) \setminus N(x) \cup \mathbf{C} \setminus N(w)$ ;
5     find induced matching of size two using Lemma 2;
6     if induced matching exists then
7       return subgraph of  $G(M)$  induced by  $x, w, u$  and the induced matching ( $G_{III_1}$ )
8     end
9   end
10  end
11  /* We can now assume that there is no  $G_{III_1}$  in  $G(M)$  */
12  set  $i_{min} = \infty$ ;
13  for  $\{x, w\} \in E_M$ , where  $x \in \mathbf{C}$  and  $w \in \mathbf{R}$  do
14    find  $D = N_2(w) \setminus N(x)$  and  $Y = N(D) \setminus N(w)$ ;
15    for  $y \in Y$  do
16      construct the subgraph  $G_{x,w,y}$  of  $G(M)$  induced by vertices  $N(w) \setminus \{x\} \cup \{y\} \cup D$ ;
17      find  $D_i = N_i(y)$  in  $G_{x,w,y}$ , for  $i \geq 1$ ;
18      find  $Y' = \{y' \in Y : D_1 \setminus N(y') \neq \emptyset\}$  and  $D' = D \cap N(Y')$ ;
19      find smallest odd  $i \geq 3$  such that  $D_i \cap D' \neq \emptyset$  (if possible);
20      if found then
21        pick any  $d_i \in D_i \cap D'$ , any  $y' \in Y' \cap N(d)$ ;
22        find a path  $P$  from  $d_i$  to some  $d_1 \in D_1$  in  $G_{x,w,y}$  of length  $i - 1$ ;
23        if  $\{y', d_1\} \notin E(M)$  and  $i < i_{min}$  then
24          set  $i_{min}$  to  $i$ ;
25          remember  $x, w, y, y'$  and vertices of  $P$ ;
26        end
27      end
28    end
29  end
30  if  $i_{min} = \infty$  then
31    return subgraph of type III not found or there is a subgraph of type I or V of the size
32    at most the size of the smallest type III subgraph
33  else
34    return subgraph of  $G(M)$  induced by remembered set of vertices
35  end

```

Correctness of Algorithm 5. It is easy to check that the first phase of the algorithm finds G_{III_1} subgraph if it exists in $G(M)$. Assume that G_{III_1} is not an induced subgraph of $G(M)$, i.e., that a smallest subgraph of type III (if it exists) has at least 9 vertices. The algorithm continues to the second phase.

First, assume that i is not found, i.e., for all odd $i \geq 3$, $D_i \cap D' = \emptyset$. This implies that any path starting at y in $G_{x,w,y}$ cannot be extended with a white vertex y' that is not adjacent to w and not adjacent to the second

vertex $d_1 \in D_1$ of this path. Hence, the algorithm correctly continues with examining another selection of vertices x, w, y . Assume that i was found. Now, assume that $G(M)$ does not contain edge $\{y', d_1\}$. Let us verify that vertices x, w, y, y' and the vertices of P induce $G_{\text{III}(i-1)/2}$. It is clear that x is connected only to w and w only to white vertices on P except the first vertex y . By the construction, each vertex on P can be adjacent only to its predecessor or successor on P . Since i is the smallest odd integer larger than two such that $D_i \cap D' \neq \emptyset$, y' is not adjacent to any black vertex on the path other than the last one. Hence, the vertices induce a subgraph of type III. Finally, assume that $\{y', d_1\} \in E(M)$. If $i \geq 5$, then vertices of P without y and y' induce a cycle of length $i + 1$, i.e., a subgraph $G_{\text{I}(i-3)/2}$, which is smaller than a subgraph of type III we could get for this selection of x, w, y (by choosing a different d_i , y' or path P , or searching for another odd i such that $D_i \cap D' \neq \emptyset$). If $i = 3$, consider $d'_1 \in D_1$ that is not adjacent to y' and let $P = y, d_1, u, d_3$. If d'_1 is adjacent to u , vertices $x, w, u, d'_1, y, d_3, y'$ induce G_{III_1} , a contradiction. Hence, assume $\{d'_1, u\} \notin E(M)$. Since $d'_1 \in D \subseteq N_2(w)$, there exists $u' \in N(w)$ adjacent to d'_1 . If $\{d_1, u'\} \in E(M)$, then vertices $x, w, u, u', d_1, d'_1, d_3, y, y'$ induce G_V . Otherwise, vertices w, u, d_1, y, d'_1, u' induce a cycle of length 6. In any case, there exists a subgraph of other type of size equal or smaller than it would be possible to find for this choice of x, w, y , hence, the algorithm correctly moves to the next choice.

Complexity of Algorithm 5. We will show that the complexity of Algorithm 5 is $O(ne^2) = O(\Delta^2 m^2)$. The body of the loop in lines 2–7 will execute $O(\Delta e)$ times and each step of the body take $O(e)$ time. Hence, the complexity of the first phase is $O(\Delta e^2) = O(ne^2)$. The main loop of the second phase will execute $O(e)$ times. Determining D and Y takes time $O(e)$. The nested loop in lines 13–26 will execute $O(n)$ times. Each step of the body of this loop will take time $O(e)$. Hence, the complexity of the second phase is $O(ne^2)$.

3.2 Type IV

Algorithm 6 finds the subgraph G_{IV} in time $O(n^3 e)$, if it exists and if it is a smallest forbidden subgraph. If there is a smaller forbidden subgraph of type I or III, it might find an instance of G_{IV} or it might report that there is a smaller forbidden subgraph instead.

Correctness of Algorithm 6. Correctness of the algorithm follows by the following lemma.

Lemma 3. *Consider a subgraph G' of $G(M)$ induced by vertices $x, y, z, u, v, w, a, b, c, d$ that contains edges*

$$\{x, a\}, \{y, b\}, \{z, c\}, \{a, u\}, \{b, v\}, \{c, w\}, \{u, d\}, \{v, d\}, \{w, d\},$$

and does not contain edges

$$\{x, d\}, \{y, d\}, \{z, d\}.$$

Then either G' is an instance of G_{IV} or G' contains either G_{I_1} , G_{III_1} or G_{III_2} as an induced subgraph.

Algorithm 6: Find a G_{IV} subgraph or report that there is a smaller subgraph of type I or III.

Input : $G(M)$
Output: A subgraph G_{IV} of $G(M)$ or report that G_{IV} is not a smallest subgraph

```

1 for distinct  $x, y, z \in \mathbf{C}$  do
2   find  $A = N(x) \setminus (N(y) \cup N(z))$ ;
3   find  $B = N(y) \setminus (N(x) \cup N(z))$ ;
4   find  $C = N(z) \setminus (N(x) \cup N(y))$ ;
5   find  $D = \mathbf{C} \setminus (N(x) \cup N(y) \cup N(z))$ ;
6   find  $U = N(A) \setminus \{x, y, z\}$ ;
7   find  $V = N(B) \setminus \{x, y, z\}$ ;
8   find  $W = N(C) \setminus \{x, y, z\}$ ;
9   if all sets  $A, B, C, D, U, V, W$  are non-empty then
10    for  $d \in D$  do
11      if there exists distinct  $u \in U \cap N(d)$ ,  $v \in V \cap N(d)$  and  $w \in W \cap N(d)$  then
12        find  $a \in A \cap N(u)$ ,  $b \in B \cap N(v)$  and  $c \in C \cap N(w)$ ;
13        if none of the edges  $\{a, v\}, \{a, w\}, \{b, u\}, \{b, w\}, \{c, u\}, \{c, v\}$  exists then
14          return  $G(M)[x, y, z, u, v, w, a, b, c, d] = G_{IV}$ 
15        else
16          return there is a smaller subgraph of type I or III
17        end
18      end
19    end
20  end
21 end
22 return not found

```

Proof. We will use the following two partial maps: $R(x) = a, R(y) = b, R(z) = c, R(a) = u, R(b) = v$ and $R(c) = w$, and $L = R^{-1}$.

If none of the edges in $E' = \{\{a, v\}, \{a, w\}, \{b, u\}, \{b, w\}, \{c, u\}, \{c, v\}\}$ is present, then G' is isomorphic to G_{IV} .

If exactly one edge e in E' is present, we have an induced subgraph G_{III_1} centered at the vertex $r = e \cap \{u, v, w\}$. In particular, vertices $d, r, L(r), L(L(r)), \ell, L(\ell), z$, where $\ell = e \cap \{a, b, c\}$ and $z \in \{u, v, w\} \setminus \{r, R(\ell)\}$, induce G_{III_1} .

We can assume that there are at least two edges in E' present. We will distinguish two cases. Either (i) there exists two edges e and e' in E' present such that $e \cap e' \neq \emptyset$, or (ii) for each pair of such edges $e \cap e' = \emptyset$.

First, consider case (i) and let e, e' be such that $e \cap e' \neq \emptyset$. Depending on whether the intersection lies in $\{a, b, c\}$ or $\{u, v, w\}$, we have two cases:

1. $e \cap e' \in \{a, b, c\}$ (“edges joining on the left”), then vertices $V(G') \setminus \{e \cap e'\}$ induce G_{III_2} ;
2. $e \cap e' \in \{u, v, w\}$ (“edges joining on the right”), then vertices $x, y, z, a, b, c, e \cap e'$ induce G_{III_1} .

Now, consider case (ii). Note the number of edges in E' present is at most three. We will consider two cases depending on the number of such edges:

1. $|E' \cap E(G')| = 2$: Without loss of generality we can assume that $e \cap \{a, b, c\} = L(e' \cap \{u, v, w\})$ for $e, e' \in E'$ present in G' . Then the same collection of vertices as in the case of one edge e induces G_{III_1} , since one end of e' lies outside of this collection.

2. $|E' \cap E(G')| = 3$: Then the vertices a, b, c, u, v, w induce C_6 , i.e., G_{I_1} .

Complexity of Algorithm 6. We will show that the complexity of Algorithm 6 is $O(n^3e) = O(\Delta mn^3)$. The first loop executes $O(n^3)$ times, determining A, B, C, D takes time $O(m)$, determining sets U, V, W time $O(e)$. The loop for $d \in D$ is executed $O(m)$ times and each execution takes time $O(\deg(d))$, i.e., the total time spent in this loop is $\sum_{d \in D} O(\deg(d)) = O(e)$.

3.3 Type V

Algorithm 7 find the subgraph G_V in time $O(n^3e)$, if it exists and if it is a smallest forbidden subgraph. If there is a smaller forbidden subgraph of type I or III, it might find an instance of G_V or it might report that there is a smaller forbidden subgraph instead.

Algorithm 7: Find a G_V subgraph or report that there is a smaller subgraph of type I or III.

```

Input :  $G(M)$ 
Output: A subgraph  $G_V$  of  $G(M)$  or report that  $G_V$  is not a smallest subgraph
1 for distinct  $x, y, z \in \mathbf{C}$  do
2   find  $A = N(x) \setminus (N(y) \cup N(z))$ ;
3   find  $B = N(y) \setminus (N(x) \cup N(z))$ ;
4   find  $C = N(z) \setminus (N(x) \cup N(y))$ ;
5   find  $D = (N(y) \cap N(z)) \setminus N(x)$ ;
6   pick any  $u \in N(A) \cap N(B) \cap N(D)$  if possible;
7   pick any  $v \in N(A) \cap N(C) \cap N(D)$  if possible;
8   if  $u$  and  $v$  has been picked then
9     if  $u \in N(C)$  or  $v \in N(B)$  then
10      | return there is a smaller subgraph of type III ( $G_{III_1}$ )
11    end
12    find  $A' = A \cap N(u) \cap N(v)$  and  $D' = D \cap N(u) \cap N(v)$ ;
13    if  $A' = \emptyset$  or  $D' = \emptyset$  then
14      | return there is a smaller subgraph of type I ( $G_{I_1}$  or  $G_{I_2}$ )
15    end
16    pick any  $a \in A', b \in B \cap N(u), c \in C \cap N(v)$  and  $d \in D'$ ;
17    return  $G(M)[x, y, z, u, v, a, b, c, d]$ 
18  end
19 end
20 return not found

```

Correctness of Algorithm 7. The algorithm is able to reduce time complexity by avoiding trying all possible choices for u, v and a, b, c, d , but rather picking one choice (if possible), and then either finding G_V or a smaller forbidden subgraph. Let us verify that decisions algorithm makes are correct:

- First, assume that the algorithm stops in line 10. Then there exists $w \in N(A) \cap N(B) \cap N(C) \cap N(D)$ (either u or v). Then there exists $a \in A \cap N(w)$, $b \in B \cap N(w)$ and $c \in C \cap N(w)$. Vertices x, y, z, a, b, c, w induce G_{III_1} .
- Assume that the algorithm stops in line 14. If $A' = \emptyset$ and $D' = \emptyset$, there exists $a \in A \cap N(u)$, $a' \in A \cap N(v)$, $d \in D \cap N(u)$ and $d' \in D \cap N(v)$. Note

that $a \neq a', d \neq d', a, d \notin N(v)$ and $a', d' \notin N(u)$. It is easy to check that vertices x, a, u, d, y, d', v, a' induce C_8 . Similarly, if either $A' = \emptyset$ or $D' = \emptyset$, we can find vertices that induce C_6 .

- Finally, it is easy to check that if the algorithm outputs an induced subgraph in line 17, it is G_V .

On the other hand, if G_V is a smallest forbidden subgraph of $G(M)$, then the algorithm cannot finish in lines 10 and 14, and hence, it will eventually output G_{IV} in line 17.

Complexity of Algorithm 7. We will show that the complexity of Algorithm 7 is $O(n^3e) = O(\Delta mn^3)$. The first loop executes $O(n^3)$ times, determining A, B, C, D takes time $O(m)$, picking u, v time $O(e)$, picking a, b, c, d time $O(m)$. Hence, the total time used by the algorithm is $O(n^3(O(m) + O(e))) = O(n^3e)$.

3.4 Main algorithm

Algorithm 8 finds a smallest forbidden subgraph using the three algorithms described above.

Algorithm 8: Find a smallest forbidden Tucker subgraph.

Input : $G(M)$
Output: A smallest forbidden subgraph of $G(M)$

```

1 find a smallest white asteroidal triple  $A$  using Lemma 1;
2 let  $\ell$  be the sum of the lengths of three paths of  $A$ ;
3 find a smallest subgraph of types III, IV and V (using the procedures described above);
4 let  $s_{III}, s_{IV}, s_V$  be the sizes of these subgraphs (or  $\infty$  if not found), respectively;
5 if  $\ell = \min\{\ell, s_{III}, s_{IV}, s_V\}$  then
6   | return  $A$ 
7 else
8   | let  $s_X = \min\{\ell, s_{III}, s_{IV}, s_V\}$ ;
9   | return the smallest subgraph of type  $X$ 
10 end
```

To verify the correctness of Algorithm 8, first consider that one of the smallest forbidden subgraphs of $G(M)$ is of type I or II. By the above argument, asteroidal triple A is of type I or II with size ℓ , and since it is a smallest forbidden subgraph, we have $\ell = \min\{\ell, s_{III}, s_{IV}, s_V\}$. Hence, the algorithm correctly outputs one of the smallest forbidden subgraphs. Second, assume that all smallest forbidden subgraphs of $G(M)$ are of type III, IV and V. Let $s = \min\{s_{III}, s_{IV}, s_V\}$. If A is of type I or II, then the size of A is ℓ , and hence, $\ell > s$ and $s_X = \min\{\ell, s_{III}, s_{IV}, s_V\}$. If A is of type III, IV or V, then $\ell \geq s + 1$, and hence again $s_X = \min\{\ell, s_{III}, s_{IV}, s_V\}$. It follows that Algorithm 8 correctly outputs one of the smallest forbidden subgraphs.

It follows from Algorithm 8 that we do not need a special detection algorithms for type I and II forbidden subgraphs. However, in some applications, there might be a need to determine a smallest forbidden subgraph of each type. Therefore, we present such algorithms for these two types of forbidden subgraphs as well.

References

1. Zaky Adam, Monique Turmel, Claude Lemieux, David Sankoff: *Common Intervals and Symmetric Difference in a Model-Free Phylogenomics, with an Application to Streptophyte Evolution*. Journal of Computational Biology 14(4): 436-445 (2007)
2. Farid Alizadeh, Richard M. Karp, Lee Aaron Newberg, Deborah K. Weisser: *Physical Mapping of Chromosomes: A Combinatorial Problem in Molecular Biology*. Algorithmica 13(1/2): 52-76 (1995)
3. Guillaume Blin and Romeo Rizzi and Stéphane Vialette: *A Faster Algorithm for Finding Minimum Tucker Submatrices*. J. Theory Comput. Syst. (51) 270-281 (2012)
4. Kellogg S. Booth and George S. Lueker: *Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms*. J. Comput. Syst. Sci. 13(3): 335-379 (1976)
5. Cedric Chauve, Eric Tannier: *A Methodological Framework for the Reconstruction of Contiguous Regions of Ancestral Genomes and Its Application to Mammalian Genomes*. PLoS Computational Biology 4(11) (2008)
6. Michael Dom and Jiong Guo and Rolf Niedermeier : *Approximation and fixed-parameter algorithms for consecutive ones submatrix problems* J.Computer and System Sciences 76 (3-4) 204-221 (2010)
7. M.Habib, Ross M. McConnell, Christophe Paul, Laurent Viennot: *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing*. Theor. Comput. Sci. 234(1-2): 59-84 (2000)
8. Wen-Lian Hsu: *A Simple Test for the Consecutive Ones Property*. J. Algorithms 43(1): 1-16 (2002)
9. Wei-Fu Lu, Wen-Lian Hsu: *A Test for the Consecutive Ones Property on Noisy Data - Application to Physical Mapping and Sequence Assembly*. Journal of Computational Biology 10(5): 709-735 (2003)
10. Nathan Lindzey and Ross M. McConnell : *On Finding Tucker Submatrices and Lekkerkerker-Boland Subgraphs*. WG 2013.
11. Jian Ma, Louxin Zhang, Bernard B. Suh, Brian J. Raney, Richard C. Burhans, W. James Kent, Mathieu Blanchette, David Haussler, and Webb Miller1 : *Reconstructing contiguous regions of an ancestral genome*. GenomeRes 16(12) 1557–1565 (2006)
12. Ross M. McConnell: *A certifying algorithm for the consecutive-ones property*. SODA 2004: 768-777
13. Joao Meidanis, Oscar Porto, Guilherme P. Telles: *On the Consecutive Ones Property*. Discrete Applied Mathematics 88(1-3): 325-354 (1998)
14. A. C. Tucker: *A structure theorem for the consecutive 1's property*. J. of Comb. Theory, Series B 12 :153-162 (1972)